딥러닝 레포트 – 1

학과 : 컴퓨터공학과　　학번 : 2019305061　　이름 : 임주형

```
# 1-1

import pandas as pd

# 데이터 파일의 URL
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv'

# CSV 파일을 읽어 데이터프레임으로 저장
red = pd.read_csv(url, sep=';')

# 데이터프레임의 처음 3개 레코드를 화면에 출력
print(red.head(3))
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  ₩
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  ₩
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
```

```
# 1-2

# 임의로 5개의 열 선택
red = red[['fixed acidity', 'volatile acidity', 'free sulfur dioxide', 'pH', 'quality']]

# 처음 5개 레코드 출력
print(red.head(5))
```

```
   fixed acidity  volatile acidity  free sulfur dioxide    pH  quality
0            7.4              0.70                 11.0  3.51        5
1            7.8              0.88                 25.0  3.20        5
2            7.8              0.76                 15.0  3.26        5
3           11.2              0.28                 17.0  3.16        6
4            7.4              0.70                 11.0  3.51        5
```

```python
# 1-3

my_name = 'LimJuHyung'

# 열 중 임의로 한 개 선택
selected_column = red.sample(1, axis=1).columns[0]

# 열 이름 변경을 위해 rename() 메서드 사용
red.rename(columns={selected_column: my_name}, inplace=True)

# 변경한 열을 index로 설정
red.set_index(my_name, inplace=True)

print(red.head())
```

```
            volatile acidity  free sulfur dioxide    pH  quality
LimJuHyung
7.4                     0.70                 11.0  3.51        5
7.8                     0.88                 25.0  3.20        5
7.8                     0.76                 15.0  3.26        5
11.2                    0.28                 17.0  3.16        6
7.4                     0.70                 11.0  3.51        5
```

```
# 2-1

import pandas as pd

# 10월 1일부터 3일까지의 핸드폰 사용 시간
cell_phone_use_time = pd.DataFrame({'10/1':[5, 2, 4, 7, 9],
                                    '10/2':[3, 6, 10, 5, 8],
                                    '10/3':[7, 2, 5, 3, 6]})

cell_phone_use_time
```

	10/1	10/2	10/3
0	5	3	7
1	2	6	2
2	4	10	5
3	7	5	3
4	9	8	6

```
# 2-2

def print_part(part, text):
    print(text)
    print(part, end='₩n₩n')

X = cell_phone_use_time.iloc[:, :-1].values   # 1일, 2일 사용시간
y = cell_phone_use_time.iloc[:, 2:].values    # 3일 사용시간

X_train = cell_phone_use_time.iloc[:3, :-1].values    # 1, 2, 3행 데이터
y_train = cell_phone_use_time.iloc[:3, -1:].values    # 1, 2, 3행 데이터
X_test = cell_phone_use_time.iloc[3:, :-1].values     # 4, 5행 데이터
y_test = cell_phone_use_time.iloc[3:, -1:].values     # 4, 5행 데이터

print_part(X_train, "1, 2일 사용시간₩n 1, 2, 3행 데이터")
print_part(y_train, "3일 사용시간₩n 1, 2, 3행 데이터")
print_part(X_test, "1, 2일 사용시간₩n 4, 5행 데이터")
print_part(y_test, "3일 사용시간₩n 4, 5행 데이터")
```

```
1, 2일 사용시간
 1, 2, 3행 데이터
[[ 5  3]
 [ 2  6]
 [ 4 10]]

3일 사용시간
 1, 2, 3행 데이터
[[7]
 [2]
 [5]]

1, 2일 사용시간
 4, 5행 데이터
[[7 5]
 [9 8]]

3일 사용시간
 4, 5행 데이터
[[3]
 [6]]
```

```
# 2-3

avg_list = []           # 평균을 담을 리스트
sum_list = [0, 0, 0]    # 합계를 담을 리스트

for index, row in cell_phone_use_time.iterrows():  # 각 행 반복
    for date, value in row.iteritems():  # 각 열 반복
        sum_list[int(date[3:]) - 1] += value   # 인덱스(날짜)에 맞게 사용시간을 추가

# 평균 구하기
for s in sum_list:
    tmp_avg = s/len(sum_list)
    avg_list.append(tmp_avg)

cell_phone_sum_avg = pd.DataFrame({'10/1':[sum_list[0], avg_list[0]],
                                   '10/2':[sum_list[1], avg_list[1]],
                                   '10/3':[sum_list[2], avg_list[2]]}, index=['sum', 'avg'])

cell_phone_sum_avg
```

```
<ipython-input-10-c195c4f3ae6e>:6: FutureWarning: iteritems is deprecated and will be removed
  for date, value in row.iteritems():  # 각 열 반복
```

	10/1	10/2	10/3
sum	27.0	32.000000	23.000000
avg	9.0	10.666667	7.666667

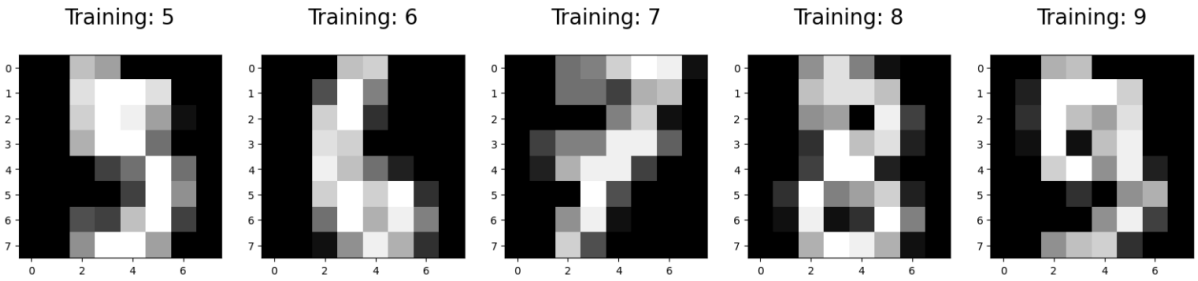
```
# 2-4

# 행과 열을 서로 변환하여 concat 시킨 후 출력
total_cell_phone_data = pd.concat([cell_phone_use_time, cell_phone_sum_avg], axis=0)
total_cell_phone_data
```

	10/1	10/2	10/3
0	5.0	3.000000	7.000000
1	2.0	6.000000	2.000000
2	4.0	10.000000	5.000000
3	7.0	5.000000	3.000000
4	9.0	8.000000	6.000000
sum	27.0	32.000000	23.000000
avg	9.0	10.666667	7.666667

```
# 3-1
%matplotlib inline
from sklearn.datasets import load_digits
digits = load_digits()
print("Image Data Shape" , digits.data.shape)
print("Label Data Shape", digits.target.shape)
```

```
Image Data Shape (1797, 64)
Label Data Shape (1797,)
```

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(digits.data[5:10], digits.target[5:10])):
    plt.subplot(1, 5, index + 1)
    plt.imshow(np.reshape(image, (8,8)), cmap=plt.cm.gray)
    plt.title('Training: %i₩n' % label, fontsize = 20)
```




```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.25, random_state=0)

from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression
LogisticRegression()

```
logisticRegr.predict(x_test[0].reshape(1,-1))
logisticRegr.predict(x_test[0:10])
```

```
array([2, 8, 2, 6, 6, 7, 1, 9, 8, 5])
```

```
predictions = logisticRegr.predict(x_test)
score = logisticRegr.score(x_test, y_test)
print(score)
```

```
0.9511111111111111
```

```
# 3-2
%matplotlib inline
from sklearn.datasets import load_digits
digits = load_digits()
print("Image Data Shape" , digits.data.shape)
print("Label Data Shape", digits.target.shape)
```

```
Image Data Shape (1797, 64)
Label Data Shape (1797,)
```

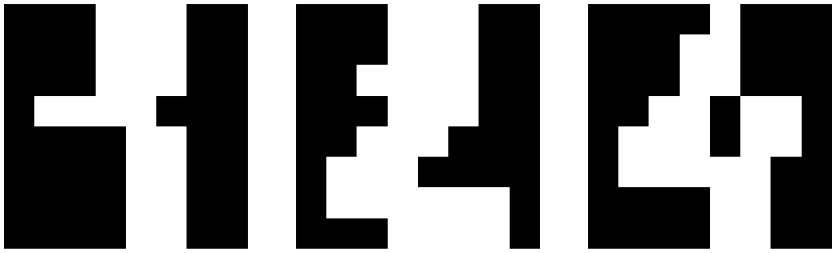
```
from google.colab import files

uploaded = files.upload()
```

파일 선택 파일 3개

- **img1.png**(image/png) - 140 bytes, last modified: 2023. 10. 1. - 100% done
- **img2.png**(image/png) - 154 bytes, last modified: 2023. 10. 1. - 100% done
- **img4.png**(image/png) - 151 bytes, last modified: 2023. 10. 1. - 100% done

```
Saving img1.png to img1.png
Saving img2.png to img2.png
Saving img4.png to img4.png
```

```
import cv2
import matplotlib.pyplot as plt
# 업로드한 이미지 경로
image_path_1 = list(uploaded.keys())[0]
image_path_2 = list(uploaded.keys())[1]
image_path_4 = list(uploaded.keys())[2]

# 이미지 전처리
img1 = cv2.imread(image_path_1)
img2 = cv2.imread(image_path_2)
img4 = cv2.imread(image_path_4)

# 흑백 변환
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
img4 = cv2.cvtColor(img4, cv2.COLOR_BGR2GRAY)

 # 크기 조정
img1 = cv2.resize(img1, (8, 8))
img2 = cv2.resize(img2, (8, 8))
img4 = cv2.resize(img4, (8, 8))

# 정규화
img1 = img1 / 255.0
img2 = img2 / 255.0
img4 = img4 / 255.0

# 이미지를 1차원 배열로 변환
img1 = img1.reshape(1, -1)
img2 = img2.reshape(1, -1)
img4 = img4.reshape(1, -1)

img_list = [img1, img2, img4]

plt.figure(figsize=(12, 4))
for i in range(len(img_list)):
    plt.subplot(1, len(img_list), i + 1)  # 1행 len(img_list)열의 subplot 생성
    plt.imshow(img_list[i].reshape(8, 8), cmap='gray')  # 이미지 출력
    plt.axis('off')  # 이미지 축 제거
plt.show()  # 모든 이미지 출력
```



```python
# 훈련과 테스트 데이터세세 분리 및 로지스틱 회귀 모델 생성
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.25, random_state=0)

from sklearn.linear_model import LogisticRegression
logisticRegr = LogisticRegression()
logisticRegr.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression
LogisticRegression()

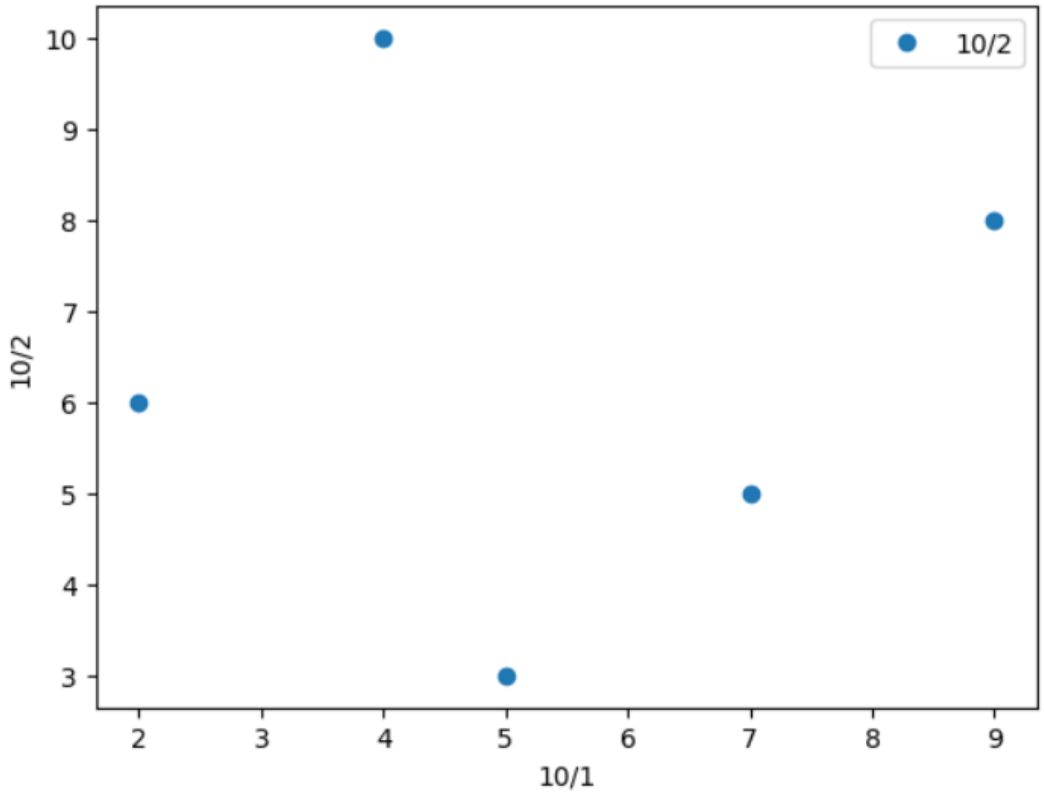
```python
from sklearn.metrics import accuracy_score

# 모델 예측
predicted_label1 = logisticRegr.predict(img1)
predicted_label2 = logisticRegr.predict(img2)
predicted_label4 = logisticRegr.predict(img4)

# 예측된 레이블 출력
print("First Image Label:", predicted_label1[0])
print("Second Image Label:", predicted_label2[0])
print("Third Image Label:", predicted_label4[0])

# 정확도 계산
predicted_list = [predicted_label1, predicted_label2, predicted_label4]
true_label = [0, 0, 0]

for i in range(3):
  for y_label in y_test:
    if predicted_list[i][0] == y_label: # 예측한 레이블과 실제 레이블이 같은지 확인
      true_label[i] =  y_label          # 같다면 정답 레이블 삽입
      break

# 3장의 이미지를 예측한 레이블과 실제 레이블을 비교하여 정확도 출력
accuracy = accuracy_score(true_label, predicted_list)
print("Accuracy:", accuracy)
```

```
First Image Label: 9
Second Image Label: 2
Third Image Label: 4
Accuracy: 1.0
```

```python
# 4-1

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics


# 10월 1일부터 3일까지의 핸드폰 사용 시간
cell_phone_use_time = pd.DataFrame({'10/1':[5, 2, 4, 7, 9],
                                    '10/2':[3, 6, 10, 5, 8],
                                    '10/3':[7, 2, 5, 3, 6]})

cell_phone_use_time
```

	10/1	10/2	10/3
0	5	3	7
1	2	6	2
2	4	10	5
3	7	5	3
4	9	8	6

```
cell_phone_use_time.plot(x='10/1', y='10/2', style='o')
plt.xlabel('10/1')
plt.ylabel('10/2')
plt.show()
```




```
# 4-3

X = cell_phone_use_time['10/1'].values.reshape(-1, 1)
y = cell_phone_use_time['10/2'].values.reshape(-1, 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8)

regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

▾ LinearRegression
LinearRegression()

```
y_pred = regressor.predict(X_test)
# df = pd.DataFrame({'Actual':y_test.flatten(), 'Predicted': y_pred.flatten()})
# df

plt.scatter(X_test, y_test, color='gray')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```